

GetExtensionVersion

Executes with elevated privileges. Use only for intended purposes.

Sean Barnum, Digital, Inc. [vita¹]

Copyright © 2007 Digital, Inc.

2007-03-23

Part "Original Digital Coding Rule in XML"

Mime-type: text/xml, size: 8473 bytes

Attack Category	• Privilege Exploitation		
Vulnerability Category	• Privilege escalation problem		
Software Context	• ISAPI		
Location	• httpext.h		
Description	<p>"Call this method from your ISAPI DLL's GetExtensionVersion entry point to initialize this object.</p> <p>This method is used to initialize the heaps, threads, and caches used by this class. It also provides the implementation of the GetExtensionVersion entry point expected by IIS."</p> <ul style="list-style-type: none">- [MSDN Article] "ATL Server Library Reference- CIIsapiExtension::GetExtensionVersion"- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vclib/html/vclrfCIIsapiExtensionGetExtensionVersion.asp <p>Don't create system resources in code that implements GetExtensionVersion as it executes with elevated security privileges (SYSTEM). Use GetExtensionVersion only for its intended purpose, which is communicating back to IIS the version of ISAPI that the extension DLL was built to support.</p>		
APIs	Function Name	Comments	
	GetExtensionVersion		
Method of Attack	Elevation of Privilege		
Exception Criteria	Unknown		
Solutions	Solution Applicability	Solution Description	Solution Efficacy
		Use the DllMain entry of your extension DLL to initialize	

1. <http://buildsecurityin.us-cert.gov/bsi-rules/35-BSI.html> (Barnum, Sean)

		resources, since it uses the security token of the client request.
SignatureDetails		<pre>CIsapiExtension::GetExtensionVersion BOOL GetExtensionVersion(HSE_VERSION_INFO* pVer) throw();</pre>
Examples of Incorrect Code		<pre>BOOL YOURCLASS::GetExtensionVersion(HSE_VERSION_INFO* pVer) { // Call default implementation for initialization Base::GetExtensionVersion(pVer); /** NOW do something ELSE, other than the intended return of . This code runs in a privileged mode and the security defaults may change */ HANDLE a = CreateThread(NULL, // default security attributes 0, ThreadProc, pData, 0, &dwThreadId[i]); //... return TRUE; }</pre>
Examples of Corrected Code		<pre>/* Copyright 1999-2004 The Apache Software Foundation * * Licensed under the Apache License, Version 2.0 (the "License"); * you may not use this file except in compliance with the License. * You may obtain a copy of the License at * * http://www.apache.org/licenses/ LICENSE-2.0 *</pre>

```

 * Unless required by applicable
law or agreed to in writing,
software
 * distributed under the License
is distributed on an "AS IS"
BASIS,
 * WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either
express or implied.
 * See the License for the
specific language governing
permissions and
 * limitations under the License.
 */
*****  

*
 * ISAPI Module Cache handling
section
*
*****  

/* Our isapi global config values
*/
static struct isapi_global_conf {
apr_pool_t *pool;
apr_thread_mutex_t *lock;
apr_hash_t *hash;
} loaded;

/* Our loaded isapi module
description structure */
struct isapi_loaded {
const char *filename;
apr_thread_rwlock_t *in_progress;
apr_status_t last_load_rv;
apr_time_t last_load_time;
apr_dso_handle_t *handle;
HSE_VERSION_INFO *isapi_version;
apr_uint32_t report_version;
apr_uint32_t timeout;
PFN_GETEXTENSIONVERSION
GetExtensionVersion;
PFN_HTTPEXTENSIONPROC
HttpExtensionProc;
PFN_TERMINATEEXTENSION
TerminateExtension;
};

static apr_status_t
isapi_unload(isapi_loaded *isa,
int force)
{
/* All done with the DLL... get
rid of it...
*

```

```

    * If optionally cached, and we
    weren't asked to force the unload,
    * pass HSE_TERM_ADVISORY_UNLOAD,
    and if it returns 1, unload,
    * otherwise, leave it alone (it
    didn't choose to cooperate.)
    */
    if (!isa->handle) {
    return APR_SUCCESS;
    }
    if (isa->TerminateExtension) {
    if (force) {
    (*isa->TerminateExtension)
    (HSE_TERM_MUST_UNLOAD);
    }
    else if (!(*isa-
    >TerminateExtension)
    (HSE_TERM_ADVISORY_UNLOAD)) {
    return APR_EGENERAL;
    }
    }
    apr_dso_unload(isa->handle);
    isa->handle = NULL;
    return APR_SUCCESS;
}

static apr_status_t
cleanup_isapi(void *isa_)
{
    isapi_loaded* isa =
    (isapi_loaded*) isa_;
    /* We must force the module to
    unload, we are about
    * to lose the isapi structure's
    allocation entirely.
    */
    return isapi_unload(isa, 1);
}

static apr_status_t
isapi_load(apr_pool_t *p,
server_rec *s, isapi_loaded *isa)
{
    apr_status_t rv;
    isa->isapi_version
    = apr_pcalloc(p,
    sizeof(HSE_VERSION_INFO));
    /* TODO: These ought to become
    overrideable, so that we
    * assure a given isapi can be
    fooled into behaving well.
    */

```

```

    * The tricky bit, they aren't
    really a per-dir sort of
    * config, they will always be
    constant across every
    * reference to the .dll no matter
    what context (vhost,
    * location, etc) they apply to.
    */
    isa->report_version = MAKELONG(0,
5); /* Revision 5.0 */
    isa->timeout = 300 * 1000000; /*  

microsecs, not used */

    rv = apr_dso_load(&isa->handle,
isa->filename, p);
    if (rv)
    {
        ap_log_error(APLOG_MARK,
APLOG_ERR, rv, s,
"ISAPI: failed to load %s", isa-
>filename);
        isa->handle = NULL;
        return rv;
    }

    rv = apr_dso_sym((void**)&isa-
>GetExtensionVersion, isa->handle,
"GetExtensionVersion");
    if (rv)
    {
        ap_log_error(APLOG_MARK,
APLOG_ERR, rv, s,
"ISAPI: missing
GetExtensionVersion() in %s",
isa->filename);
        apr_dso_unload(isa->handle);
        isa->handle = NULL;
        return rv;
    }

    rv = apr_dso_sym((void**)&isa-
>HttpExtensionProc, isa->handle,
"HttpExtensionProc");
    if (rv)
    {
        ap_log_error(APLOG_MARK,
APLOG_ERR, rv, s,
"ISAPI: missing
HttpExtensionProc() in %s",
isa->filename);
        apr_dso_unload(isa->handle);
        isa->handle = NULL;
        return rv;
    }

```

```

/* TerminateExtension() is an
optional interface */
rv = apr_dso_sym((void**)&isa-
>TerminateExtension, isa->handle,
"TerminateExtension");
SetLastError(0);

/* Run GetExtensionVersion() */
if (!(isa->GetExtensionVersion)
(isa->isapi_version)) {
apr_status_t rv =
apr_get_os_error();
ap_log_error(APLOG_MARK,
APLOG_ERR, rv, s,
"ISAPI: failed call to
GetExtensionVersion() in %s",
isa->filename);
apr_dso_unload(isa->handle);
isa->handle = NULL;
return rv;
}

```

Source Reference	http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vclib/html/vclrfCIsapiExtensionGetExtensionVersion.asp ²	
Recommended Resource		
Discriminant Set	Operating System • Windows	Languages • C • C++

Cigital, Inc. Copyright

Copyright © Cigital, Inc. 2005-2007. Cigital retains copyrights to this material.

Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and “No Warranty” statements are included with all reproductions and derivative works.

For information regarding external or commercial use of copyrighted materials owned by Cigital, including information about “Fair Use,” contact Cigital at copyright@digital.com¹.

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

1. <mailto:copyright@digital.com>